An Energy-Efficient Low-Voltage Swing Transceiver for mW-Range IoT End-Nodes

Hayate Okuhara*, Ahmed Elnaqib*, Davide Rossi*, Alfio Di Mauro‡, Philipp Mayer‡, Pierpaolo Palestri†, Luca Benini*‡
*DEI, University of Bologna, Italy
†DPIA, University of Udine, Italy
‡ Integrated System Laboratory ETH, Zuerich, Switzerland

***Abstract*—As the Internet-of-Things (IoT) applications become more and more pervasive, IoT end nodes are requiring more and more computational power within a few mW of power envelope, coupled with high-speed and energy-efficient inter-chip communication to deal with the growing input/output and memory bandwidth for emerging near-sensor analytics applications. While traditional interfaces such as SPI cannot cope with these tight requirements, low-voltage swing transceivers can tackle this challenge thanks to their capability to achieve several Gbps of bandwidth at extremely low power. However, recent research on high-speed serial links addressed this challenge only partially, proposing only partial or stand-alone designs, and not addressing their integration in real systems and the related implications. In this paper, we present for the first time a complete design and system-level architecture of a low-voltage swing transceiver integrated within a low-power (mW range) IoT end-node processors, and we compare it with existing microcontroller interfaces. The transceiver, implemented in a commercial 65-nm CMOS technology achieves 10.2x higher energy efficiency at 15.7x higher performance than traditional microcontroller peripherals (single lane).**

***Index Terms*—IoT, SerDes, Energy efficient peripheral, SPI, microcontroller.**

I. Introduction

Pushed by the IoT trends, in the last years, the required computational performance in end-nodes has increased considerably. Nowadays, near-sensor applications, such as convolutional neural network (CNN) based image analysis and bio-potential processing, have to efficiently operate on large volumes of sensor data captured by microcontrollers (MCUs). In this scenario, state of the art SoCs have already achieved performance in the order of several GOPS within a 10mW power envelope [1], [2].

On the other hand, in modern embedded systems operating in the IoT context, overcoming the limitations imposed by low chip-to-chip communication bandwidths represents a major challenge. Conventional MCU peripherals, such as I2C, I2S, and SPI provide transfer data rates in the order of few tenths of Mbps, which are typically not sufficient to satisfy the expected bandwidth and energy efficiency demand of the next-generation IoT applications. For example, according to the results reported in [21], the off-chip bandwidth required to perform MobileNetV2 inference [22] at 10 FPS on an MCU is larger than 500 Mbps. Although there are some solutions which can reach this requirement (e.g. HyperBus or Octal SPI operating at fast frequencies) [3], [18], their power consumption rapidly saturates the end-node power budgets.

Serial links peripherals [6]–[8], [12], [19], relying on analog data transceivers, constitute a promising alternative to purely digital serial interfaces, both from the bandwidth and energy efficiency perspective. In serial links, serialized data are sent at high rate, while low-power consumption is guaranteed by exploiting low-voltage swing signals at the physical layer. State of the art solutions [7], [8] can achieve over 1 Gbps bandwidth, while keeping the power consumption in a few mW ranges.

TABLE I
Reported low power serial links and our system

Reported work	[12]	[7], [8]	[19]	This work
Target bandwidth (Gbps)	1	1-6	25	0.8
Power consumption (mW)	< 1	< 4	29.25/pin	4.5
Additional external voltage source	No	Required	Required	No
System integration	No	No	No	Yes
Maturity	Only circuit simulation	Silicon	Silicon	Post-layout simulation

While various research efforts have been reported in optimizing serial links, system-level integration, e.g. in microcontrollers, has not been extensively studied to the authors' best knowledge. Also, in the IoT context, it is essential to minimize the data transmission power to not erode the available power budget dedicated to useful computation. Table I provides an overview of recent research efforts on low-power transceivers, positioning the proposed work with respect to state of the art transceivers in terms of power and bandwidth, and highlighting the limitations of the latter with respect to system level integration issues. This includes the need for several external power supplies forming and additional source of power consumption not considered in previous works.

From the observations above, the contributions of this paper are as follows:

- We designed a serializer-deserializer link (SerDes) system and we integrated it into an open-source low-power microcontroller [9]. Detailed architectural and micro-architectural information are shown.
- We evaluated the energy efficiency of the implemented SerDes with post-layout simulations. This brings guidelines for its power management.
- We explored a duty-cycled operation of the SerDes for a low bandwidth target. We report on the trade-off between bandwidth and energy efficiency.

The energy efficiency of the SerDes was finally compared with conventional digital peripherals widely adopted in microcontrollers such as SPI [2], [4], [5] and more advanced peripherals such as HyperBus [3]. The SerDes achieves 10.2x higher energy efficiency at 787 Mbps than the case of a Single SPI operating at 50 Mbps. Moreover, even if we target a low bandwidth such as 10 Mbps with the SerDes, its efficiency is 8.3x higher than the SPI. Also, we show the SerDes energy is 21x smaller than the Hyper Bus.

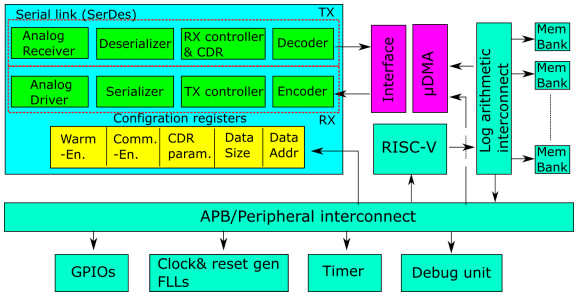

Fig. 1. High level architectural block diagram of the system overview

II. System Overview

Fig. 1 shows an overview of the System on a Chip (SoC) hosting the proposed serial link. The main building blocks of the SoC are a RISC-V core coupled to a multi-bank word-level interleaved memory, and an autonomous input/output subsystem (μDMA) [10] to transfer data to the peripherals. The internal clock is generated by a frequency locked loops (FLL). Additionally, the SoC features a timer, a debug unit, and programmable GPIOs. The SerDes is composed of the transmitter (TX), the receiver (RX), and configuration registers mapped on the advanced peripheral bus (APB) used to access enable signals, as well as the address, and the size of the communicated data. The SerDes is connected to the μDMA, an autonomous DMA subsystem providing high-speed data transfers between L2 and the peripherals.

Data from the μDMA are transmitted to another chip via the TX module. Its enable signals ("*Comm-En*" and "*Warm-En*") are from memory-mapped registers that are accessed via software. The transferred data is captured by the RX module and delivered to the μDMA. The μDMA sends the received data to the RX buffer which is allocated in the global memory according to the configuration registers.

The SerDes operates in three modes: idle, warm-up, and data-comm. During the idle mode, all the digital circuits are deactivated, and the transceiver is in low-power mode. The data-comm mode sends/receives serialized data. However, to establish a communication, the RX has to be synchronized with the transmitted data generated by another chip, potentially operating at a different clock phase. Hence, during the warm-up mode, the TX sends a training sequence including its clock phase information to the RX. According to this input, the RX recovers the transmitter clock. These three modes are selected through "*Comm-En*" and "*Warm-En*" registers.

To start the actual inter-chip communication, the TX is firstly set to the warm-up mode. The RX in another chip receives this information through a GPIO, resulting in the RX warm-up mode as well. Using the timer in Fig. 1, the processor in the RX chip waits a fixed amount of time until the RX clock is ready for the communication. Then, through a GPIO, the RX chip notifies the TX chip that the clock is ready. Also, the information required by the μDMA is stored in the configuration registers. When this is finished, through another GPIO, the RX also informs the TX that the data communication is ready. Finally, the SerDes mode is changed to data-comm mode, and the TX starts to send main data by declaring its start and end point with a communication header (Start flit), and a footer (Stop flit).

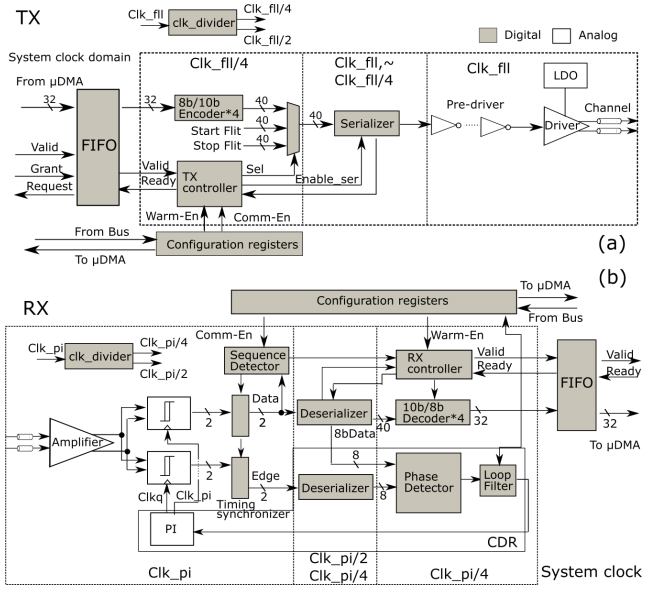

Fig. 2. Architectural block diagram of the serial link (a)TX (b)RX

III. Low-power Serial Link

A. Link architecture

Fig. 2 shows a detailed block diagram of the SerDes. The TX is composed of an 8b/10b encoders, TX controller, 40:1 serializer, pre-driver, and the driver. The RX is equipped with the analog comparators, timing synchronizers, a deserializer, RX controller, Clock Data Recovery (CDR) circuit, and the 10b/8b decoders. Both the TX and RX operate at the same frequency. However, as previously described, the clock phase in the RX has to be adjusted. The CDR circuit performs the clock recovery so that the RX clock transitions occur at the mid-point of the received data bit. The data communication is conducted by a differential signal. Hence, four analog pads are required in addition to three GPIOs used to synchronize the RX and TX in different chips.

B. TX design

At the TX, 40-bit "Start flit", "Stop flit", and the main body of the communication are serialized and transmitted to the RX in another chip. The multiplexer in Fig. 2 (a) selects one of them and sends it to the serializer which output serialized data at the double data rate (DDR) of the TX clock. Then, the driver transmits the data to the RX with low-voltage swing (200mV) signals. Here we adopt the serializer and driver in [12].

The main body of the communication is encoded by the four parallel 8b/10b encoders [11] which ensure that the serialized data is DC-balanced and its disparity is less than ± 2. The TX controller is a finite state machine that manages the timing of these functionalities according to the FIFO handshaking signals from the interface between the SerDes and the μDMA, and the enable signals from the configuration registers. The TX clock is provided by the FLL and divided by two and four. "*Clk_fll/4*" is utilized for the encoders, multiplexer, and controller to reduce the power consumption. Since the μDMA operates at the system clock, the interface between the SerDes and μDMA is implemented by an asynchronous FIFO.

Firstly, the TX is set to the idle state by the state machine of the TX controller. By asserting "*Warm-En*", its state is changed to the warm-up mode which outputs a training sequence generated by the encoders. "Start flit" is sent when the transferred

data is ready (*Valid*="1") and "*Comm-En*" is asserted. After the header is transferred, the state is automatically changed to the data-comm mode, then the main part of the data communication is started. During this mode, the input of the serializer is updated every 20 cycles as the serial data are synchronized at DDR. When the "*Valid*" signal is negated, "Stop flit" is sent. Finally, the state is back to the idle one.

C. RX design

At the RX, the input is firstly captured by the analog comparators [15] which restore the even and odd bit data from the channel. These bits are buffered by the timing synchronizers, then deserialized, decoded, and sent to the μDMA through the asynchronous FIFO interface. We employ the deserializer architecture reported in [12]. The "timing synchronizers" here are buffers to ensure the timing constraints between digital and analog circuits.

Since the data communication begins from "Start flit" and ends at "Stop flit", the sequence detector monitors whether or not they arrive. This is realized by checking 11011111 ($K_{27,7}$ in [11]) for "Start flit" and 10111111 ($K_{29,7}$) for "Stop flit". According to the information from the detector, the RX controller manages the deserializer and 10b/8b decoders for the main body of the transferred data. The decoded data with the "*Valid*" signal is sent to the FIFO when its "*Ready*" is asserted.

The generated clock by the CDR scheme is divided into four ("*Clk_pi/4*") and two ("*Clk_pi/2*"). The RX controller, decoders, and some parts of the CDR loop are synchronized at "*Clk_pi/4*" to reduce the power consumption. *Clk_pi/2* is utilized by the deserializer.

1) Sequence detector: In the sequence detector, the even and odd bits captured by the analog comparators are checked to activate the entire RX when the start flit arrives. The detector is composed of a finite state machine as shown in Fig. 3. The state of the detector changes when the $K_{27,7}$ arrives. In other words, when the first two bit of 11011111 (i.e. 11) is detected, the next state is "Check1". After this, if the following two bits are 01, the state is updated to "Check2". When all the bits of $K_{27,7}$ are detected, the deserializer and decoder are enabled through the RX controller. Also, during the data communication, it is monitored whether or not the stop flit arrives with a similar procedure. When this is detected, the state of the detector is backed to "Start".

It is important to mention that the RX has to consider whether or not a bit shift occurs at arriving data. In other words, even if a bit is sent as even bit at the TX side, there is no guarantee that it is captured as even bit at the RX. For example, the sequence of 11011111 might be captured as x1 10 11 11 1x. To manage this, the state machine holds the bit shift information as the signal "*Shift*". Since an additional 2 bits have to be checked when "*Shift*" is asserted, the "Check4" state is implemented.

Also, note that the timing synchronizer adjusts the bit shift according to the "*Shift*" signal from the sequence detector after detecting the start flit. Hence, the deserializer always receives the even and odd bit correctly.

2) RX controller: During the warm-up mode, the controller activates only the parts of the CDR loop. After the loop is settled, an enable signal for the sequence detector is provided from the configuration registers. When the start flit arrives, the controller state is in the data-comm mode which enables the entire deserializer. The decoders update their output when

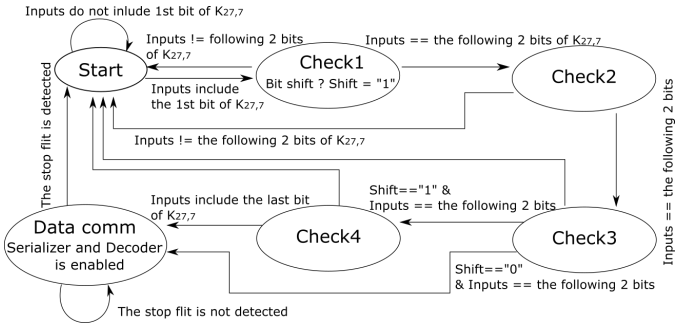

Fig. 3. State machine of the sequence detector

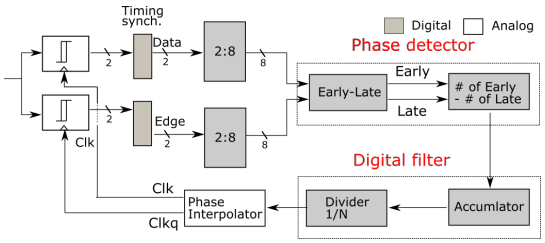

Fig. 4. Architectural diagram of the CDR loop

40bit data is ready. This timing is notified by the deserializer. The "*Valid*" signal is also generated after the latency of the decoders. When the stop flit arrives, the controller disables the 8:40 deserializer and decoders if "*Warm-En*" is still asserted. In case that all the enable signals for the RX are negated, the RX is in the idle mode.

3) Clock Data Recovery module: The CDR scheme is composed of the phase detector, digital filter and phase interpolator (PI) which adjusts the phase of the FLL clock (Fig. 4). The "Early-Late" module consists of seven parallel Alexander phase detectors [14] that compares 8-bit"*Data*" captured by the normal clock ("*Clk*") with 8-bit "*Edge*" synchronized at a quadrature clock ("*Clkq*"). Then, the number of "*Early*" is subtracted by the number of "*Late*". The result is accumulated and divided by 1/N (N=1,2,4,8,..., 128) at every 4 clock cycles. According to the divider output, the PI shifts the clock phase for both of "*Clk*" and "*Clkq*". The resolution of this adjustment is set to $2\pi/32$ in the current design. The PI is a charge-based interpolator based on [16].

IV. Implementation

We implemented a system-level layout including the SerDes. A 65-nm bulk CMOS technology [17] was used. This design includes three FLLs [13] as clock generators and 128KB of the L2 bank. Two of the FLLs are for the microcontroller and peripherals except for the SerDes. The last one is dedicated to the link for a testing purpose. At actual systems, one of the other FLLs is shared with the SerDes to save the system power consumption. The analog signals are connected to 4 library I/O cells featuring a built-in 50-ohm resistor. Two of them are for the RX and the rests are for the TX. Synopsys Design Compiler 2018.06-SP1 and Cadence Innovous v15.20 were employed for the synthesis and P&R.

The nominal voltage and operational frequency of the SerDes are 1.2V and 400MHz, respectively. Hence, the target bandwidth of the current design is 0.8 Gbps as the data transfer is performed at DDR. Also, 1.2V is used for both digital and analog circuits. This is because adding another voltage source increases system costs which should be avoided for embedded microcontrollers.

TABLE II
POWER CONSUMPTION OF THE SERDES @ 1.2V

Power consumption (Analog parts)	RX	2.85mW
	TX	0.59mW
Power consumption (Digital parts)	RX	0.591mW : data-comm mode 0.367mW : warm-up mode 0.433μW : idle mode
	TX	0.239 mW :data-comm & warm-up 32.7μW : idle

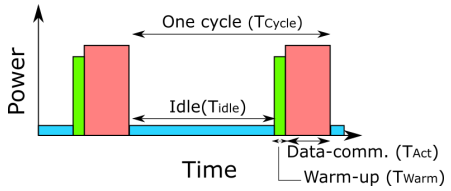

Fig. 5. Conceptual timing diagram of the duty-cycled operation

V. RESULTS

To evaluate the energy efficiency of the proposed SerDes system, post-layout simulations are conducted with Synopsys Prime Time M-2016.12-SP3 for the digital part and Cadence Spectre 6.1 for the analog part. Table II shows the estimated power consumption at 1.2V of V_{DD} and 400MHz of operational frequency. Since the TX power is dominated by the analog part and the serializer, other parts are omitted.

According to the results, the entire power consumption of the SerDes is 4.27mW when the serial link is in the data-comm mode. The energy efficiency of the implemented link is 5.34pJ/bit. A power of 4.05 mW is consumed during the warm-up mode because most of the RX components need to be activated. If the analog parts are turned off via an off-chip power switch during the idle state, the entire link power is 33.1 μW.

In case that a required bandwidth is lower than 0.8Gbps, the power consumption is further lowered. However, since the CDR loop is designed for 0.8Gbps, lowering its operational frequency causes a loop convergence problem. Instead, a duty-cycled operation [20] which periodically turns on the SerDes is adopted in this paper. Fig. 5 shows its conceptual timing diagram. Here, $T_{Cycle}, T_{Act}, T_{Warm}$ and T_{Idle} represent one cycle period, duration of the data-comm, warm-up, and idle mode, respectively. The data communication is conducted until the RX buffer in the global memory is filled up. Then, the link state is back to the idle mode. When it is activated again, the warm-up mode settles the CDR loop with the overhead of T_{Warm}.

Using these assumptions and the values in Table II, the SerDes energy efficiency during the duty-cycled operation is obtained (see Fig. 6). For a comparison to other existing peripherals, this graph also depicts the read/write average energy consumption of a single SPI (40-nm) and Hyper Bus (65-nm) implementation with an I/O voltage of 1.8V. The transferred data size of the Hyper Bus was 0.5 KB. The Hyper Bus is implemented by fast but power-hungry drivers, while the SPI adopts slow but low power ones. Hence, the SPI and Hyper Bus operate up to 50 and 100MHz, respectively. In other words, the maximum bandwidth of the former and latter are 50 Mbps and 1.6Gbps. As can be seen from the graph, the Hyper Bus consumes much higher energy than the single SPI due to the I/O drivers even though the Hyper Bus achieves a bandwidth over 1Gbps. Thus, at the conventional digital interfaces, there is a trade-off between the maximum bandwidth and energy efficiency. On the other hand, our SerDes achieves

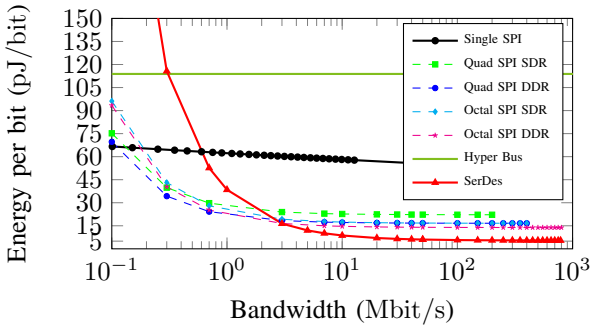

Fig. 6. Energy consumption compared to other peripherals

TABLE III
THE NUMBER OF DATA PADS NEEDED FOR EACH SOLUTION

Single SPI	Quad SPI	Octal SPI	Hyper Bus	This work
4	6	11	12	4

a high bandwidth and low energy consumption simultaneously. Indeed, the maximum bandwidth (BW_{max}) with the 16KB RX buffer is 787Mbps. Compared to the best case of the Single SPI (i.e. at 50Mbps), the SerDes efficiency is 10.2x higher at 15.7x higher performance. Besides, even if the target bandwidth is lowered to 10Mbps, the proposed SerDes achieves 8.3x smaller energy than the SPI. Moreover, although the Hyper Bus achieves about 2 times higher bandwidth, its energy efficiency is 21x lower than our SerDes operating at BW_{max}.

Based on the SPI measurement results (Fig. 6) and its switching activity, we estimated the energy efficiency of a Quad-SPI and Octal-SPI operating at both DDR and SDR which are also shown in Fig. 6. As can be seen from the graph, the parallel SPI lanes improve the energy efficiency, at the cost of additional overheads in terms of pad usage (Table III), which is critical for small and often pad limited microcontrollers. Nevertheless, the proposed SerDes still achieves lower energy consumption, at a 3x smaller pad area cost. Indeed, the SerDes energy efficiency at BW_{max} is 2.56x higher than the case of the DDR Octal SPI, joining the benefits of low pad frame overhead, high bandwidth and high energy efficiency, essential features for next-generation near-sensor data analytics low-power architectures.

VI. CONCLUSION

In this paper, we presented the system architecture of a high-speed/low-power serial link. The proposed SerDes simultaneously provides a high bandwidth and energy efficiency for embedded systems, unlike traditional digital interfaces such as SPIs and a Hyper Bus. The evaluation results showed that, thanks to the low-voltage swing property, the SerDes achieves about 10.2x higher energy efficiency at 15.7x higher bandwidth than the Single SPI link. Also, the duty-cycled operation allows the SerDes to achieve 8.3x higher energy efficiency than the Single SPI even at 10Mbps, a low bandwidth requirement. Moreover, when compared to the Hyper Bus, the SerDes energy is 21x smaller.

ACKNOWLEDGMENT

This work was supported in part by the WiPLASH (Architecting More Than Moore – Wireless Plasticity for Heterogeneous Massive Computer Architectures) project founded from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 863337.

REFERENCES

[1] D. Rossi et al., "Energy-Efficient Near-Threshold Parallel Computing: The PULPv2 Cluster," in IEEE Micro, vol. 37, no. 5, pp. 20-31, September/October 2017.

[2] A. Pullini, D. Rossi, I. Loi, G. Tagliavini and L. Benini, "Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing," in IEEE Journal of Solid-State Circuits, vol. 54, no. 7, pp. 1970-1981, July 2019.

[3] Cypress Semiconductors, HyperRAM Memory, Available Online: http://www.cypress.com/products/hyperram-memory, (last checked 07/30/2019).

[4] Y. Pu et al., "A 9-mm2 Ultra-Low-Power Highly Integrated 28-nm CMOS SoC for Internet of Things," in IEEE Journal of Solid-State Circuits, vol. 53, no. 3, pp. 936-948, March 2018.

[5] T. Karnik et al., "A cm-scale self-powered intelligent and secure IoT edge mote featuring an ultra-low-power SoC in 14nm tri-gate CMOS," 2018 IEEE International Solid - State Circuits Conference - (ISSCC), San Francisco, CA, 2018, pp. 46-48.

[6] B. Razavi, "Historical Trends in Wireline Communications: 60? Improvement in Speed in 20 Years," in IEEE Solid-State Circuits Magazine, vol. 7, no. 4, pp. 42-46, Fall 2015.

[7] W. Choi et al., "3.8 A 0.45-to-0.7V 1-to-6Gb/S 0.29-to-0.58pJ/b source-synchronous transceiver using automatic phase calibration in 65nm CMOS," 2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers, San Francisco, CA, 2015, pp. 1-3.

[8] W. Choi et al., "A 0.45-0.7 V 1-6 Gb/s 0.29-0.58 pJ/b Source-Synchronous Transceiver Using Near-Threshold Operation," in IEEE Journal of Solid-State Circuits, vol. 53, no. 3, pp. 884-895, March 2018.

[9] PULP-platform, PULPissimo, Available Online: https://github.com/pulp-platform/pulpissimo, (last checked 07/30/2019).

[10] A. Pullini, D. Rossi, G. Haugou and L. Benini, "μDMA: An autonomous I/O subsystem for IoT end-nodes," 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Thessaloniki, 2017, pp. 1-8.

[11] A. X. Widmer, P. A. Franaszek, "A DC-balanced, partitioned-block, 8B/10B transmission code", IBM Journal of Research and Development, v.27 n.5, p.440-451, September 1983.

[12] M. Dazzi et al., "Sub-mW multi-Gbps chip-to-chip communication Links for Ultra-Low Power IoT end-nodes," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, 2018, pp. 1-5.

[13] D. Bellasi, P. Schönle, Q. Huang and L. Benini, "A wide tuning-range ADFLL for mW-SoCs with dithering-enhanced accuracy in 65 nm CMOS," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, 2017, pp. 1-4.

[14] B. Razavi, "Challenges in the design high-speed clock and data recovery circuits," in IEEE Communications Magazine, vol. 40, no. 8, pp. 94-101, Aug. 2002.

[15] S. Babayan-Mashhadi and R. Lotfi, "Analysis and Design of a Low-Voltage Low-Power Double-Tail Comparator," in IEEE Transactions on VLSI Systems, vol. 22, no. 2, pp. 343-352, Feb. 2014.

[16] A. Elnaqib and S. A. Ibrahim, "Low-power charge-steering phase interpolator," in Electronics Letters, vol. 52, no. 10, pp. 810-812, 2016.

[17] UMC, 65nm technology. Available Online: http://www.umc.com/english/pdf/UMC%2065nm.pdf, June 2005. (last checked:09/05/2019)

[18] AP Memory Technology Corp., IoT RAM & ADMUX PSRAM. Available Online: http://www.apmemory.com/html/product_psram.php. (last checked:09/05/2019)

[19] J. W. Poulton et al., "A 1.17-pJ/b, 25-Gb/s/pin Ground-Referenced Single-Ended Serial Link for Off- and On-Package Communication Using a Process- and Temperature-Adaptive Voltage Regulator," in IEEE Journal of Solid-State Circuits, vol. 54, no. 1, pp. 43-54, Jan. 2019.

[20] T. Anand, A. Elshazly, M. Talegaonkar, B. Young and P. K. Hanumolu, "A 5 Gb/s, 10 ns Power-On-Time, 36 μW Off-State Power, Fast Power-On Transmitter for Energy Proportional Links," in IEEE Journal of Solid-State Circuits, vol. 49, no. 10, pp. 2243-2258, Oct. 2014.

[21] A. Burrello, F. Conti, A. Garofalo, D. Rossi, and L. Benini, "Work-in-Progress: DORY: Lightweight Memory HierarchyManagement for Deep NN Inference on IoT Endnodes", International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2019), pp.1-2.

[22] Sandler, Mark et al., "Mobilenetv2: Inverted residuals and linear bottlenecks", 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510–4520.